PVFS BOF Session

Details of PVFS2

PVFS2 Development Team

The PVFS Project

- PVFS project started in mid-1990s at Clemson
  - Code in use today still has pieces from 1995
  - We've been pushing that code well beyond its original design constraints
- Meanwhile we've been working with application groups, other researchers, other file systems, high-level I/O libraries
  - Better perspective on the field as a whole
  - More insight into capabilities of systems, potential pitfalls

A New PFS: PVFS2

- Obviously a new design was in order
  - Embodying the principles we feel are key for performance and usability
  - Supporting the hardware available today
  - Scaling to expected system sizes
- PVFS2 is this new design
  - *Intended to quickly mature into a production-quality PFS*
  - Also a basis for research into next-generation systems (autonomous storage, etc.)
- Collaboration between ANL and Clemson, plus others

Collaborators

- Northwestern I/O group
  - A. Choudhary, W. Liao, A. Ching, J. Li, K. Coloma
- Ohio State Supercomputer Center
  - P. Wyckoff, T. Baer
- Ohio State University
  - D.K. Panda, J. Wu

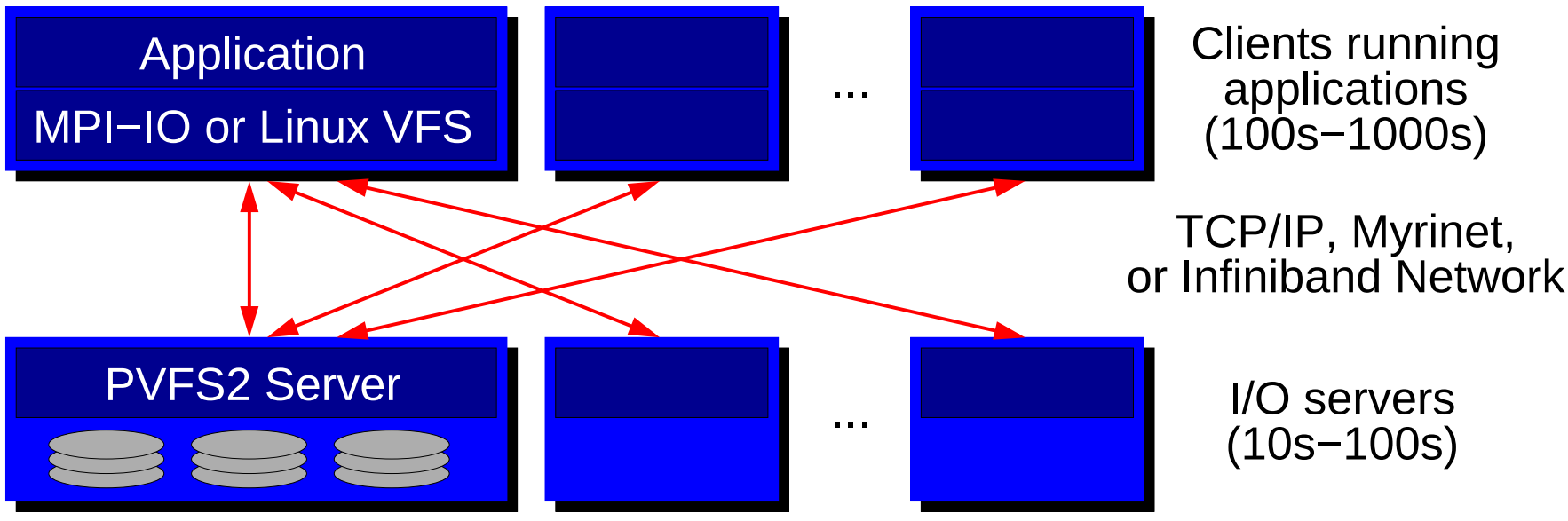
PVFS2 Status

- First public release was made available earlier in the week
- Web site, mailing lists, documentation, CVS access all in place
- Everything can be found off http://www.pvfs.org/pvfs2

System Requirements

- Servers
  - Linux (2.4 or 2.6) IA32, IA64, Alpha, PPC
  - Local disk (ext2, ext3, xfs, reiserfs, etc.)
  - TCP, Myrinet, or InfiniBand network
- Clients
  - VFS for Linux (2.6 only right now), IA32 tested (others in theory)
  - MPI-IO on Linux (2.4 or 2.6) IA32, IA64, Alpha
- Ready for beta testing, not for production use

PVFS2 at a Glance



- “Intelligent” servers with PFS-oriented protocol
- Messaging over existing communication network
- Storage on locally attached disks (possibly shared for failover purposes)

Design Goals

- Careful to limit the scope of the effort
- *Effective solution for parallel applications*
- Must concentrate on
  - Performance
  - Reliability
  - Maintainability
  - Ease of Administration
- Other applications are fair game
  - No design changes that compromise parallel, scientific I/O usability

Performance

- Extracting performance from underlying hardware through appropriate abstractions
- Key workloads/access patterns
  - File manipulation (e.g. `ls`, `cp`)
  - Application parallel I/O (e.g. MPI-IO, HDF5, PnetCDF)
  - *IOzone and Bonnie++ are not useful parallel I/O benchmarks*
- Optimizations focus on these two types of access
- Consistency semantics tuned to match

Reliability

- HW components are fault prone
- To address this
  - Minimize impact of faults on system as a whole
  - Leverage existing (proven) failover solutions
  - Keep the approach simple
- A stateless system made up of independent entities seems best *if it will meet the other needs*
  - Fewer cascading failures
  - Simplifies fault handling
  - Standard HA approaches are applicable

Maintainability

- Keeping components in user space when possible facilitates debugging, portability
- Intermediate languages to manage complexity (e.g. state machines)
- *Concurrent access* test suite being built alongside system
  - Automate testing for correctness and performance
  - Eliminate re-introduction of bugs

Ease of Administration

- Protocol support for maintenance and monitoring operations
  - Iterating through objects
  - Event logging
  - Performance logging
- Tools for examining the system
  - Visualizing file system layout, run-time performance

PVFS2 Servers

- Single server type handles both metadata and I/O operations
- Servers are independent, only communicate with clients
- Distributed metadata fully supported, no restrictions
  - Some optimizations not possible with multiple metadata servers
  - But load is spread across multiple servers
- Servers store data on locally accessible storage

Networks

- PVFS2 is designed to support multiple networks
  - Message-oriented, reliable, ordered network abstraction (BMI)
  - Implementations for TCP, InfiniBand, and Myrinet GM
    - Thank Pete Wyckoff for IB
- Multiple network types may be in use for a single FS
- Multi-homing of servers not yet in place, but planned

Application Interfaces

- Two methods of interface are supported:
  - MPI-IO – parallel applications
  - Mountable file system – utilities and administration
- Client libraries are provided for research purposes and for building administration tools
  - Tailored for supporting MPI-IO and VFS operations
  - Not for casual users (doesn't look like POSIX)

Storage

- Multiple storage targets are possible
- Single implementation at this time
  - Berkeley DB used for metadata and directory storage
    - Convenient iterators, key searches
  - UNIX files used for data storage
    - Obvious choice for streams of bytes
- Leverages AIO calls as appropriate
- Explicit caching and O_DIRECT use in progress

Summary

- PVFS2 is now available for people to try
- We're focusing our efforts
  - Parallel, scientific application performance
  - Statelessness for superior fault handling
  - Management tools
  - Testing to guarantee reliability of implementation
- Everything is available at http://www.pvfs.org
- We'd like to stop here and answer your questions